Finite Difference Derivatives
CLA 365 TECHNICAL REPORT

Charles Zhang Emil Bao Le Henry Bell Zuofu Huang

February 18, 2020

Abstract

In this report, we present R code that implement methods for approximating the derivatives of continuous functions. We implement function D using the forward difference method. Based on function D, we update our Newton's method so that the derivative is computed within Newton's method instead of inputted. We then discuss the drawbacks of the forward-difference method and produce a comparison between forward-difference, backwards difference and three-point centered-difference method.

Introduction

The derivative is widely used in exploring the global and local behavior of functions. We implement finite difference methods to numerically find derivatives in R. We also investigate δ values that minimize errors and perform cross-method comparisons among forward difference, backward difference and three-point centered difference methods.

1 Derivative Function

1.1 Overview

Our derivative function is based on the method of finite differences. The method of finite differences builds on the concept that the slope is $\frac{\Delta y}{\Delta x}$. Assuming f is continuous, $\frac{\Delta y}{\Delta x}$ can be approximated by $\frac{f(x_+\delta)-f(x)}{\delta}$ where $\delta \in \mathbb{R}$. The accuracy of our approximation of the slope (denoted f') improves as $\delta \to 0$. This is only one of three finite difference methods, the others will be discussed later in the paper. This one is called the forward method.

$$f'(x) = \lim_{\delta \to 0} \frac{f(x-\delta) - f(x)}{\delta} \tag{1}$$

1.2 Implementation

```
1 D <- function(f, delta=.001) {
2   result <- function(x) (f(x+delta)-f(x))/delta
3   return(result)
4 }
```

1.3 Testing

We tested our derivative on a function whose derivative is known: $f(x) = x \cdot e^{-x^2}$. Figure 1 plots the original function $f(x)$.Figure 2 plots the function's derivative computed with function D.

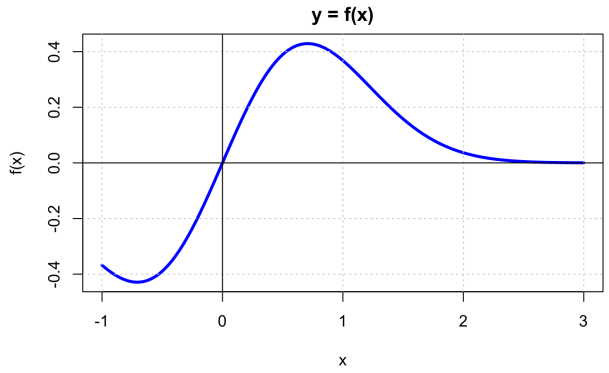


Figure 1: $f(x) = xe^{-x^2}$

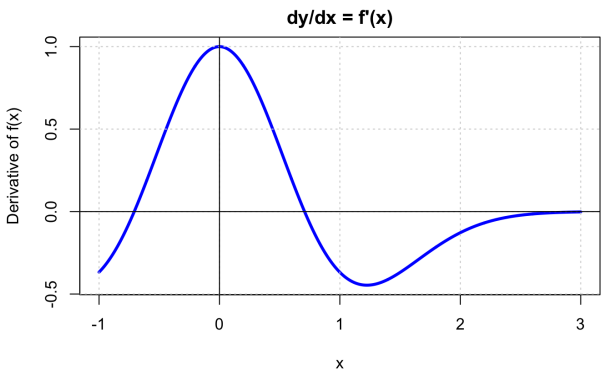


Figure 2: $f'(x)$ computed by our function D

2 Newton's Method

2.1 Overview

The Derivative function can be applied to Newton's method, which utilizes the derivative of a function to compute its root. Newton's method starts with an initial guess, finds the tangent line to the function at that point and then computes the root of that line. It uses this root as the next guess and repeats this process until a maximum number of iterations is reached or the change between iterations is less than the tolerance.

2.2 Implementation

```
Newton <- function(f, x_0, num.its = 40,tol=0.5*10^-10) {
  history = rep(NA, num.its + 1)
  history[1] = x_0
  i = 0
  fprime = D(f)
  while (i < num.its ) {
    i = i + 1
    x_0 = x_0 - f(x_0)/fprime(x_0)
    history[i+1] = x_0
    if (abs(history[i + 1] - abs(history[i])) < tol)
      break
  }
  return(list(root=x_0,history=history[!is.na(history)]))
}
```

2.3 Testing

We test the improved Newton's method code on the function $f(x) = \cos(x)^2 + \sin(x) - 1$ with the root at $x = \pi$. With an initial guess of 4, our newton's method converges super-linearly to the root π in 7 iterations. This root has an error of 7.1×10^{-15}, which is on the order of machine epsilon.

For some functions, however, Newton's method does not converge. An example of this is the cycling example $f(x) = 4x^4 - 6x^2 - 11/4$, whose graph is shown below in Figure 3. With an initial guess of $\frac{1}{2}$, Newton's method will bounce between $\frac{1}{2}$ and $-\frac{1}{2}$, and will never converge to a root.

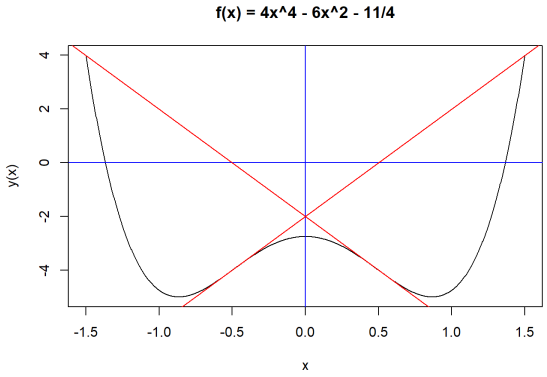


Figure 3: Newton Method Test with $f(x) = 4x^4 - 6x^2 - 11/4$.

3 Optimizing δ Value

3.1 Overview

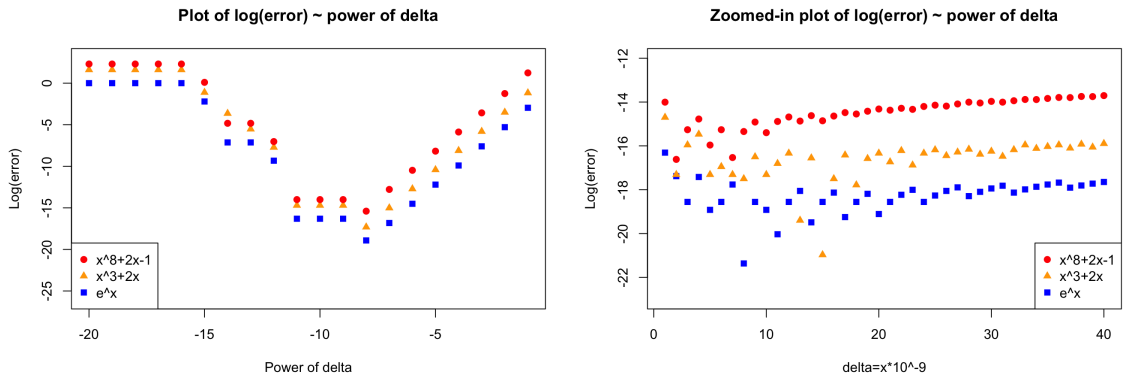
Theoretically, the finite difference equation is most accurate when δ approaches zero. However, this proves to be computationally infeasible because we cannot divide by zero. For this reason, it must be asked: which value of δ corresponds to the most accurate derivative?

3.2 Implementation

The procedure we take to examine which δ is the most appropriate consists of: 1. picking a function 2. applying our derivative method to this function with varying δ values. We then computed the error of each δ and find out which one yields the least error. We perform this procedure on three functions: $f(x) = e^x$, $g(x) = x^3 + 2x$ and $h(x) = x^8 + 2x - 1$. Our results can be seen in figure 4.

```
f <- function(x) {exp(x)}
slopes=function(f,x){
  df <- D(f, delta = 10^(-1:-20))
  return(df(x))
}
slopes <- slopes(f,0)  # slope should be 1
error <- abs(slopes - 1)
```

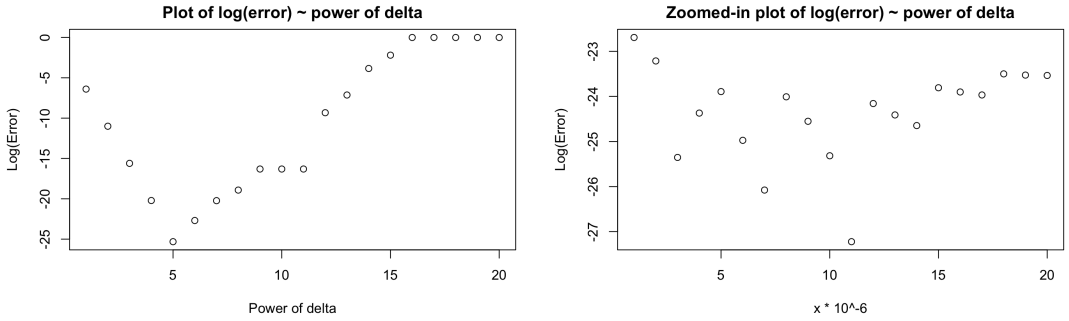
In figure 4a, the log of the error starts to decrease for the three functions as the power of δ decreases from -1 before it reaches ≈ -8 (indicating $\delta = 10^{-8}$). Then the error starts to increase again. We zoomed in on the behavior of δ near 10^{-8}(shown in figure 4b). From this we concur that the optimal δ is dependent on the function, but is around $\delta = 10^{-8}$.



(a) (b)

Figure 4: log error of δ values for three functions. (a) Behavior of error as the power of δ decreases. (b) Zoomed-in plot of (a)

Interesting to notice, δ is in the neighborhood of 10^{-8} for all 12 polynomial functions we tried. However, δ behaves differently for trigonometric functions such as $\sin(x)$. See Figure 5.



(a) (b)

Figure 5: log error of δ values for $\sin(x)$. (a) Behavior of error as the power of δ decreases. (b) Zoomed-in plot of (a) yields 1.1×10^{-5}.

The code below shows how the error for e^x with regards to different values of δ is generated. The process for three functions is analogous. See Table 1: delta and error values where the error is minimized.

```
df <- D(f, delta = ((.5+(1:100000)/100000))*10^-8)
error <- abs(df(0)-1)
data.frame("delta"=c(((.5+(1:100000)/100000))*10^-8), "error"=e) %>%
  arrange(e) %>%
  head()
```

e^x		$x^3 + 2x$		$\sin(x)$	
δ	*error*	δ	*error*	δ	*error*
8.80230×10^{-9}	8.382184×10^{-14}	1.18080×10^{-8}	8.633094×10^{-13}	1.04631×10^{-5}	6.661338×10^{-16}
1.27741×10^{-9}	1.136868×10^{-13}	1.15932×10^{-8}	3.028688×10^{-12}	1.10002×10^{-5}	6.661338×10^{-16}
1.36328×10^{-8}	2.687850×10^{-13}	1.35254×10^{-8}	3.029577×10^{-12}	5.68430×10^{-6}	1.110223×10^{-15}
1.19154×10^{-8}	5.513368×10^{-13}	1.00906×10^{-8}	6.081358×10^{-12}	9.48030×10^{-6}	1.443290×10^{-15}
7.94360×10^{-9}	5.515588×10^{-13}	1.32037×10^{-8}	7.414513×10^{-12}	8.88960×10^{-6}	1.554312×10^{-15}

Table 1: Optimized δ for three functions correct to 5 decimal places. Left to right: e^x, $x^3 + 2x$, $\sin(x)$.

3.3 Result

Our primary conclusion is that $\delta \approx 10^{-8}$ for polynomial functions, but δ fluctuates depending on the function it is used on. More specifically, δ increases to approximately 10^{-5} for trigonometric functions such as $\sin(x)$. Although theoretically $E(f'(x+\delta) - f'(x)) \to 0$ as $\delta \to 0$, such behavior of δ can be considered as a drawback of numerical analysis.

4 Differences between finite difference methods

4.1 Overview

The finite difference method comes in multiple forms, the forward equation which is featured in section 1 (1.1), the three-point centered difference (4.1), and the backward difference method (4.2).

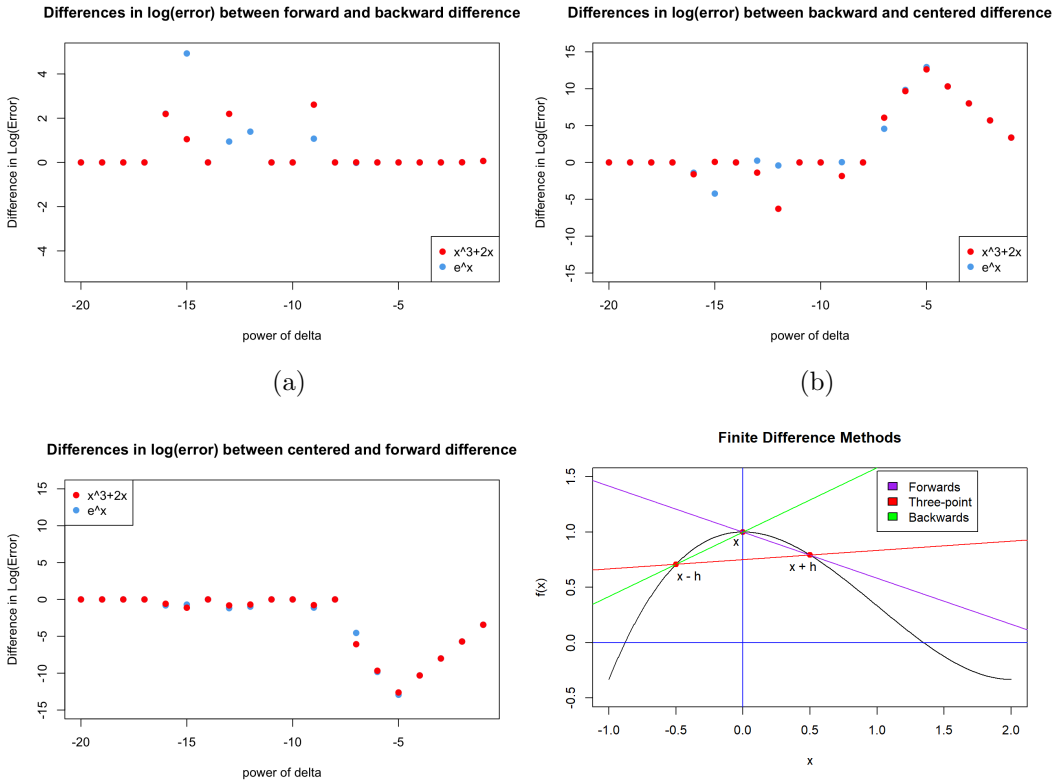
$$f'(x) = \lim_{\delta \to 0} \frac{f(x-\delta) - f(x+\delta)}{2\delta} \qquad (4.1) \qquad\qquad f'(x) = \lim_{\delta \to 0} \frac{f(x) - f(x-\delta)}{\delta} \qquad (4.2)$$

A visual representation of these equations on an arbitrary function can be found in Figure 6d on the next page. The three methods differ on which points they compute the slope between. The forward difference method uses points x and x + δ, the centered method uses x - δ and x + δ and the backwards method uses x - δ and x.

4.2 Testing

In order to test which method produces the least error, we choose two functions, e^x and $x^3 + 2x$. We apply each method with different δ values and subtract the pairwise error. A positive difference leads to a worse performance. Therefore, we can conclude that the forward method has the worst performance. See Figure 6.

It is worth noting that while centered difference method performs better as $\delta \to 10^{-8}$, it performs worse as the power decreases from 10^{-8}.



(a) (b)

(c)

(d) Visualization representation of three finite difference methods

Figure 6: Pairwise difference in log(error)

Conclusion

Theoretically, we should have more precise results from the different methods of differentiation by decreasing δ. However, the opposite is true due to rounding errors. As rounding errors are inevitable in numerical differentiation, there is a threshold for how small δ can get depending on the input functions we choose. In this report, we have shown that the smallest δ value for the least errors for e^x, $x^3 + 2x$ and sin(x) are, respectively, 8.80230×10^{-8}, 1.18080×10^{-8} and 1.04631×10^{-5}.

Furthermore, the graphs of log(errors) for these 3 functions affirm our prediction that as δ exceeds the threshold respectively, error starts to increase. By plotting the difference errors resulted from these two methods to δ value, we also found that the three-point centered-difference method provides more precise result compared with the two-point forward-difference method. This can be explained by that δ can be approximately as small as the cubic root of $\epsilon_{\text{machine}}$ for the first method, while it can only be as small as the square root for $\epsilon_{\text{machine}}$ for the latter. Due to this reason, for future work we can explore how threshold for δ of three-point centered-difference method to find the characteristics of a function that determine its optimal δ value.